LETTER Communicated by Shimon Edelman

Learning to Recognize Three-Dimensional Objects

Dan Roth
danr@uiuc.edu
Ming-Hsuan Yang
myang1@uiuc.edu
Narendra Ahuja
n-ahuja@uiuc.edu
Department of Computer Science and the Beckman Institute
Urbana, IL 61801, U.S.A.

A learning account for the problem of object recognition is developed within the probably approximately correct (PAC) model of learnability. The key assumption underlying this work is that objects can be recognized (or discriminated) using simple representations in terms of syntactically simple relations over the raw image. Although the potential number of these simple relations could be huge, only a few of them are actually present in each observed image, and a fairly small number of those observed are relevant to discriminating an object.

We show that these properties can be exploited to yield an efficient learning approach in terms of sample and computational complexity within the PAC model. No assumptions are needed on the distribution of the observed objects, and the learning performance is quantified relative to its experience. Most important, the success of learning an object representation is naturally tied to the ability to represent it as a function of some intermediate representations extracted from the image.

We evaluate this approach in a large-scale experimental study in which the SNoW learning architecture is used to learn representations for the 100 objects in the Columbia Object Image Library. Experimental results exhibit good generalization and robustness properties of the SNoW-based method relative to other approaches. SNoW's recognition rate degrades more gracefully when the training data contains fewer views, and it shows similar behavior in some preliminary experiments with partially occluded objects.

1 Introduction

The role of learning in computer vision research has become increasingly significant. Statistical learning theory has had an influence on many appli-

cations: classification and object recognition, grouping and segmentation, illumination modeling, scene reconstruction, and others. The rising role of learning methods, made possible by significant improvements in computing power and storage, is largely motivated by the realization that explicit modeling of complex phenomena in a messy world cannot be done without a significant role of learning. Learning is required for model and knowledge acquisition, as well as to support generalization and avoid brittleness. However, many statistical and probabilistic learning methods require making explicit assumptions, for example, on the distribution that governs the occurrences of instances in the world. For visual inference problems such as recognition, categorization, and detection, making these assumptions seems unrealistic.

This work develops a distribution-free learning theory account to an archetypical visual recognition problem: object recognition. The problem is viewed as that of learning a representation of an object that, given a new image, is used to identify the target object in it. The learning account is developed within the probably approximately correct (PAC) model of learnability (Valiant, 1984). This framework allows us to quantify success relative to the experience with previously observed objects, without making assumptions on the distribution, and study the theoretical limits of what can be learned from images in terms of the expressivity of the intermediate representation used by the learning process. That is, learnability guarantees that objects sampled from the same distribution as the one that governs the experience of the learner are very likely to be recognized correctly. In addition, the framework gives guidelines to developing practical algorithmic solutions to the problem.

Earlier work discussed the possibility of identifying the theoretical limits of what can be learned from images (Shvaytser, 1990) and found that learning in terms of the raw representation of the images is computationally intractable. Attempts to explain this focused on the dependence of learnability on the representation of the object (Edelman, 1993) but failed to provide a satisfactory explanation for it or a practical solution.

The approach developed here builds on suggestions made in Kushilevitz and Roth (1996) and relies heavily on the development of a feature-efficient learning approach (Littlestone, 1988; Roth, 1998; Carlson, Cumby, Rosen, & Roth, 1999). This is a learning process capable of learning quickly and efficiently in domains in which the number of potential features is very large, but any concept of interest actually depends on a fairly small number of them. At the heart of the learning approach are two assumptions that we abstract as follows:

Representational: There exists a (possibly infinite) collection $\mathcal{M}$ of "explanations" such that an object o can be represented as a simple function of elements in $\mathcal{M}$.

Procedural: There exists a process that given an image in which the target object o occurs, efficiently generates "explanations" in $\mathcal{M}$ that are present in the image and such that, with high probability, at least one of them is in the representation of o.

Under these assumptions, we prove that there exists an efficient algorithm that, given a collection of images labeled as positive or negative examples of the target object, can learn a good representation of the object. That is, it can learn a representation that with high probability would make correct predictions on future images that contain (or do not contain) the object. Furthermore, we show that under these conditions, the learned representations are robust under realistic noisy conditions. A significant nonassumption of our approach is that it has no prior knowledge of the distribution of images nor it is trying to estimate it. The learned representation is guaranteed to perform well when tested on images sampled from the distribution[1] that governed the data observed in its learning experience. Section 2 describes this framework in detail.

The framework developed here is very general. The explanations alluded to above can represent a variety of computational processes and information sources that operate on the image. They can depend on local properties of the image, the relative positions of primitives in the image, and even external information sources or context variables. Thus, the theoretical support given here applies also to an intermediate learning stage in a hierarchical process. In order to generate the explanations efficiently, this work assumes that they are syntactically simple in terms of the raw image. However, the explanation might as well be syntactically simple in terms of previously learned or inferred predicates, giving rise to hierarchical representations. The main assumptions of the framework are discussed in section 3, where we also provide some concrete examples to the type of representations used.

For this framework to contribute to a practical solution, there needs to be a computational approach that is able to learn efficiently (in terms of both computation and number of examples) in the presence of a large number of potential explanations. Our evaluation of the theoretical framework makes use of the SNoW learning architecture (Roth, 1998; Carlson et al., 1999) that is tailored for these kind of tasks. The SNoW system (available online at http://L2R.cs.uiuc.edu/~cogcomp/) is described in section 4. This is followed by a comparison of SNoW with support vector machines in section 5.

In section 6, we review and compare our method with previous works on view-based object recognition. We then describe our experiments comparing SNoW to other approaches in section 7. In these experiments, SNoW

[1] It will be clear from the technical discussion that the distribution is over the intermediate representation (features) generated given the images.

is shown to exhibit a high level of recognition and robustness. We find that the SNoW-based approach compares favorably with other approaches and behaves more robustly in the presence of fewer views in the training data. We conclude with some directions for future work in section 8.

2 Learning Framework

We study learnability within the standard PAC model (Valiant, 1984) and the mistake-bound model (Littlestone, 1988). Both learning models assume the existence of a concept class $\mathcal{C}$, a class of $\{0, 1\}$-valued functions over an instance space X with an associated complexity parameter (typically, X's dimensionality) n, and some unknown target concept $f_T \in \mathcal{C}$ that we are trying to learn. In the mistake-bound model, an example $x \in X$ is presented at each learning stage; the learning algorithm is asked to predict $f_T(x)$ and is then told whether the prediction was correct. Each time the learning algorithm makes an incorrect prediction, we charge it one mistake. We say that $\mathcal{C}$ is mistake-bound learnable if there exists a polynomial-time prediction algorithm $\mathcal{A}$ (possibly randomized) that for all $f_T \in \mathcal{C}$ and any sequence of examples is guaranteed to make at most polynomially many (in n) mistakes. We say that $\mathcal{C}$ is expected mistake-bound learnable if there exists $\mathcal{A}$, as above, such that the expected number of mistakes it makes for all $f_T \in \mathcal{C}$ and any sequence of examples is at most polynomially many (in n). Note that the expectation is taken over the random choices made by $\mathcal{A}$; no probability distribution is associated with the sequences. In learning an unknown target function $f_T \in \mathcal{C}$ in the PAC model, we assume that there is a fixed but arbitrary and unknown distribution D over the instance space X. The learning algorithm sees examples drawn independently according to D together with their labels (positive or negative). Then it is required to predict the value of f_T on another example drawn according to D. Denote by $h(x)$ the prediction of the algorithm on the example $x \in X$. The error of the algorithm with respect to f_T and D is measured by $error(h) = Pr_{x \in D}\{f_T(x) \neq h(x)\}$.

We say that $\mathcal{C}$ is PAC learnable if there exists a polynomial time learning algorithm $\mathcal{A}$ and a polynomial $p(\cdot, \cdot, \cdot)$ such that for all $n \geq 1$, all target concepts $f_T \in \mathcal{C}$, all distributions D over X, and all $\epsilon > 0$ and $0 < \delta \leq 1$, if the algorithm $\mathcal{A}$ is given $p(n, 1/\epsilon, 1/\delta)$ examples, then with probability at least $1 - \delta$, $\mathcal{A}$'s hypothesis, h, satisfies $error(h) \leq \epsilon$. It can be shown that if a concept class $\mathcal{C}$ is learnable in the expected mistake-bound model (and thus in the mistake-bound model), then it is PAC learnable (Haussler, Kearns, Littlestone, & Warmuth, 1988).

The agnostic learning model (Haussler, 1992; Kearns, Schapire, & Sellie, 1994), a variant of the PAC learning model, might be more relevant to practical learning; it applies when one does not want to assume that the labeled training examples (x, l) arise from a target concept of an a priori known structure $f_T \in \mathcal{C}$. In this model, one assumes that data elements (x, l) are sampled according to some arbitrary distribution D on $X \times \{0, 1\}$. D may

simply reflect the distribution of the data as they occur "in nature" without assuming that the labels are generated according to some "rule." As in the PAC model, the goal is to find an approximately correct h in some class $\mathcal{H}$ of hypotheses. In terms of generalization bounds, the models are similar, and therefore we will discuss the PAC/mistake-bound case here.

In practice, learning is done on a set of training examples, and its performance is then evaluated on a set of previously unseen examples. The hope that a classifier learned on a training set will perform well on previously unseen examples is based on the basic theorem of learning theory (Valiant, 1984; Vapnik, 1995); stated informally, it guarantees that if the training data and the test data are sampled from the same distribution, good performance on large enough training sample guarantees good performance on the test data (implying good "true" error),[2] where the difference between the performance on the training data and that on the test data is parameterized using a parameter that measures the richness of the hypothesis class $\mathcal{H}$. For completeness, we simply cite the following uniform convergence result:

Theorem 1 (Haussler, 1992). *If the size of the training sample S is at least*

$$m(\epsilon, \delta) = \frac{1}{\epsilon^2}\left(kVC(\mathcal{H}) + \ln\frac{1}{\epsilon} + \ln\frac{1}{\delta}\right),$$

then with probability at least $1 - \delta$, the learned hypothesis $h \in \mathcal{H}$ satisfies

$$error_D(h) < error_S(h) + \epsilon,$$

where k is some constant and $VC(\mathcal{H})$ is the VC dimension of the class $\mathcal{H}$ (Vapnik, 1982), a combinatorial parameter that measures the richness of $\mathcal{H}$.

2.1 Learning Scenario. Let $\mathcal{I}$ be an input space of images. Our goal is to learn a definition such as $\texttt{apple}:\mathcal{I} \rightarrow \{0, 1\}$ that when evaluated on a given image, outputs 1 when there is an apple in the image and 0 otherwise. It is clear, though, that this target function is very complex in terms of the input space; in particular, it may depend on relational information and even quantified predicates. Many years of research in learning theory, however, have shown that efficient learnability of complex functions is not feasible (Angluin, 1992). In the learning scenario described here, therefore, learning will not take effect directly in terms of the raw input. Rather, we will learn the target definitions in terms of an intermediate representation that will

[2] In this sense, the evaluation done in section 7 after training on a small training set is not as optimal as, say, a face detection study (Yang, Roth, & Ahuja, 2000c) done on a large training set. However, the theory quantifies the dependence of the performance on the size of the training data, and the experimental study exhibits how different algorithmic approaches fare with a relatively small number of examples.

be generated from the input image. This will allow us to learn a simpler functional description, quantifying learnability in terms of the expressivity of the intermediate representation as well as the function learned on top of it; in particular, it would make explicit the requirements from an intermediate representation so that learning is possible.

A relation[3] over the instance space $\mathcal{I}$ is a function $\chi: \mathcal{I} \rightarrow \{0, 1\}$. χ can be viewed as an indicator function over $\mathcal{I}$, defining the subset of those elements mapped to 1 by χ. A relation χ is active in $I \in \mathcal{I}$ if $\chi(I) = 1$.

Given an instance, we would like to transform it so that it is represented as the collection of the relations that are active in it. We would like to do that, though, without the need to write down explicitly all possible relations that could be active in the domain ahead of time. This is important, in particular, over infinite domains or in on-line situations where the domain elements are not known in advance and therefore it is impossible to write down all possible relations. An efficient way to do that is given by the construct of relation-generating functions (Cumby & Roth, 2000).

Definition 1. *Let $\mathcal{X}$ be an enumerable collection of relations over $\mathcal{I}$. A relation generation function (RGF) is a mapping G: $\mathcal{I} \rightarrow 2^{\mathcal{X}}$ that maps $I \in \mathcal{I}$ to a set of all elements in $\mathcal{X}$ that satisfy $\chi(I) = 1$. If there is no $\chi \in \mathcal{X}$ for which $\chi(I) = 1$, $G(I) = \phi$.*

RGFs can be thought as a way to define kinds of relations, or to parameterize over a large space of relations. Only when presented with an instance $I \in \mathcal{I}$ is a concrete relation generated. For example, G may be the RGF that corresponds to all vertical edges of size 3 in an image. Given an image, a list of all these edges that are present in the image is produced. It will be clear in section 2.3 that RGFs can be noisy to a certain extent since the framework can tolerate some amount of noise at the features level.

2.2 Learning Approach. We now present a mistake-bound algorithm for a class of functions that can be represented as *DNF* formulas over the space $\mathcal{X}$ of all relations. As indicated, this implies a PAC learning algorithm, but the proof for the mistake-bound case is simpler. In section 7, we will learn a more general function—a linear threshold function over conjunctions of relations in $\mathcal{X}$. We discuss later how the theoretical results can be expanded to this case.

Definition 2. *Let $\mathcal{X}$ be a set of relations that can be generated by a set of RGFs. Let $\mathcal{M}$ be a collection of monomials (conjunctions) over the elements of $\mathcal{X}$ and*

[3] In the machine learning literature, a relation is sometimes called a feature. We use the term *relation* here to emphasize that it is a Boolean predicate and that, in principle, it could be a higher-order predicate, that is, it could take variables (Cumby & Roth, 2000). In this article, features are simple functions (e.g., monomials) over relations.

$p(n)$, $q(n)$, and $g(n)$ be polynomials. Let $\mathcal{C}_M$ be the class of all functions that are disjunctions of at most $p(n)$ monomials in $\mathcal{M}$. Following Kushilevitz and Roth (1996), we call $\mathcal{C}_M$ polynomially explainable *if there exists an efficient (polynomial time) algorithm $\mathcal{B}$ such that for every function $f \in \mathcal{C}_M$ and every positive example of f as input, $\mathcal{B}$ outputs at most $q(n)$ monomials (not necessarily all of them are in $\mathcal{M}$) such that with probability at least $1/g(n)$, at least one of them appears in f (the probability is taken over the coin flips of the (possibly probabilistic) algorithm $\mathcal{B}$).*

It should be clear that the class of polynomially explainable DNFs is a strict generalization over the class of, say, k-DNF, for any fixed k. This difference might be important in the current context. It might be possible that some structural constraints govern the generation of the monomials, placing it in $\mathcal{M}$, but the size of the monomials is not fixed. As a canonical example for the difference, consider the following example.

Example 1. Let $S_1, \ldots, S_t$ be subsets of $\{x_1, \ldots, x_n\}$, where t is polynomial in n. Let $\mathcal{M}$ be any collection of monomials with the property that for every $m \in \mathcal{M}$, the set of variables in m is S_i for some $1 \leq i \leq t$ (i.e., any set S_i may correspond to at most $2^{|S_i|}$ monomials in $\mathcal{M}$, by choosing for each $x_j \in S_i$ whether x_j or $\bar{x}_j$ appears in the monomial). If there exists an efficient algorithm $\mathcal{B}$ that on input n enumerates these sets (and possibly some more), then $\mathcal{C}_M$ is polynomially explainable. The reason is that although $\mathcal{M}$ may contain an exponential number of monomials, given an example $(x_1, \ldots, x_n)$, only one element in each of the S_is might correspond to it. That is, the data-driven nature of the process allows working with families of features that could be very large, provided that only a reasonable number of them (polynomial, in our definition) is active in each observation.

As a concrete instantiation, consider the class of all DNF formulas in which the variables in each monomial have consecutive indices—for example, $f = x_1\bar{x}_2x_3x_4 \vee \bar{x}_4\bar{x}_5x_6 \vee x_8x_9$. Clearly, $f \notin k$-DNF, for any constant k. However, it is easy to enumerate the $\binom{n}{2} < n^2$ sets $S_{i,j}$ $(1 \leq i \leq j \leq n)$ defined by $S_{i,j} = \{x_i, x_{i+1}, \ldots, x_j\}$ and, as above, although the number of corresponding monomials is exponential, given an example, only one is relevant in each $S_{i,j}$. A similar situation occurs when learning representations of visual objects. In this case, the sets might be defined by structural constraints, and each pixel will take a constant number of values (albeit larger than 2, as in the above examples) but only one of these will be relevant given an example.

We note that in principle, it is possible to abstract the generation of the conjunctions into the RGFs (see definition 1). However, we would like to emphasize the generation of conjunctions over simple relations and the possibility of learning on top of it, given arguments in the literature of its

effectiveness and potential biological plausibility (Fleuret & Geman, 1999; Ullman & Soloviev, 1999). The elements generated by the algorithm $\mathcal{B}$, the explanations, are what we will later call the "features" supplied to the learning algorithm.

Definition 2 implements the assumptions that we abstracted in section 1. The algorithm $\mathcal{B}$ is the procedural part. $\mathcal{B}$ keeps a small set of syntactically simple definitions, the RGFs, and given an image it outputs those instantiations that are present in the image. All we require here is that with non-negligible probability, it outputs at least one "relevant" explanation (and possibly many that are irrelevant). The class $\mathcal{C}_M$ implements our representational assumption; we assume that each object has a simple (disjunctive) representation over the relational monomials. This assumption can be verified only experimentally, as we do later in this article.

We emphasize that f itself is *not* given to the algorithm $\mathcal{B}$. Also note that a function f in the class $\mathcal{C}_M$ may have few equivalent representations as a disjunction of monomials in $\mathcal{M}$. The definition requires the output of the algorithm $\mathcal{B}$ to satisfy the above property, independent of which of these representations of f are considered. The importance of this will become clear when we analyze the learning algorithm below.

Theorem 2. *If $\mathcal{C}_M$ is polynomially explainable, then $\mathcal{C}_M$ is expected mistake bound learnable. Furthermore, if $\mathcal{C}_M$ is polynomially explainable by an algorithm $\mathcal{B}$ that always outputs at least one term of f (i.e., $g(n) \equiv 1$), then $\mathcal{C}_M$ is mistake bound learnable.*

Proof. The algorithm is similar to an algorithm presented in Blum (1992), which learns a disjunction in the infinite attribute model. The algorithm maintains a hypothesis h, which is a disjunction of monomials. Initially, h contains no monomials (i.e., $h \equiv FALSE$). Upon receiving an example e, the algorithm predicts $h(e)$; if the prediction is correct, h is not updated. Otherwise, upon a mistaken prediction, it proceeds as follows:

- If e is positive: Execute $\mathcal{B}$ (the algorithm guaranteed by the assumption that $\mathcal{C}_M$ is polynomially explainable) on the example e, and add the monomials it outputs to h.
- If e is negative: Remove from the hypothesis h all the monomials that are satisfied by e (there must be at least one).

The analysis of the algorithm is straightforward for the case $g(n) \equiv 1$ and more subtle in general. To analyze the algorithm, we first fix a representation for f as a disjunction of monomials in $\mathcal{M}$ (in case f has more than one possible representation, choose one arbitrarily; we can work with any representation of f that uses only monomials in $\mathcal{M}$). Now note that an active monomial (i.e., a monomial that appears in this representation of the target

function f) is never removed from h. Therefore, since on a positive example e, the algorithm $\mathcal{B}$ is guaranteed to output at least one monomial that appears in f with probability at least $1/g(n)$, then the expected number of mistakes made on positive examples is at most $p(n) \cdot g(n)$. This also implies that the expected total number of monomials included in h during the execution of the algorithm is not more than $p(n) \cdot q(n) \cdot g(n)$.[4] Each mistake on a negative example results in removing at least one of the monomials included in h but not in f. The expected number of these monomials is therefore at most $p(n) \cdot q(n) \cdot g(n)$. The expected total number of mistakes made by the algorithm is $O(p(n) \cdot q(n) \cdot g(n))$. Finally, note that in the case $g(n) \equiv 1$, we get a truly mistake-bound algorithm, whose number of mistakes is bounded by $p(n) \cdot q(n)$.

The algorithm used in practice, in SNoW, is conceptually similar. The main difference is that the hypothesis h used is a general linear threshold function over elements in $\mathcal{M}$ rather than a disjunction, which is a restricted linear threshold function. Algorithmically, rather than dropping elements from it, their weights are updated. The details of this process (see section 4) are crucial for our approach to be computationally feasible for large-scale domains and for robustness. In order to expand the theoretical results to this case, we appeal to the results in Littlestone (1988), modified to the case of the infinite attribute domain. If the target function is indeed a disjunction over elements in $\mathcal{M}$, our results hold directly, with a much improved mistake bound that depends mostly on the number of elements in $\mathcal{M}$ that are actually relevant to f. If f is a general linear function over $\mathcal{M}$, this behavior still holds with an additional dependence on the margin between positive and negative examples, as measured in the $\mathcal{M}$ space (δ, in Littlestone, 1988; see details there).

2.3 Robustness. Any realistic learning framework needs to support different kinds of noise in its input. Several kinds of noise have been studied in the literature in the context of PAC learning, and algorithms of the type we consider here have been shown to be robust to them. The most studied type of noise is that of classification noise (Kearns & Li, 1993) in which the examples are assumed to be given to the learning algorithm with labels that are flipped with some probability, smaller than 1/2. Learning in our framework can be shown to be robust to this kind of noise, as well as to a more realistic case of attribute noise, in which the description of the input itself is corrupted to a certain degree. We believe that this is the type of noise that is more relevant in the current case. First, learning is done in terms of

[4] Note that if $\mathcal{B}$ was guaranteed only to give a monomial that appears in some representation of f, then this bound is false (as it could be the case that the active monomials in different executions of $\mathcal{B}$ belong to different representations of f). This explains the requirement of the definition that seems too strong.

the output of the RGFs, which may introduce some noise. Second, attribute noise is related to occlusion noise, which is important in object recognition. Specifically, attribute noise can be used to model the type of noise that usually occurs when other objects appear in the image, behind or in front of the target object. This is formalized next using the notion of domination.

Let f_1, f_2 be two concepts. We say that f_1 is k-dominated by f_2 if each f_1 example can be obtained from an f_2 example by flipping the (binary) value of at most k of the active relations. In this case, f_2 k-dominates f_1. The labels of the examples, however, are generated according to the original concept, before the noise is introduced.

Theorem 3. *If a class $\mathcal{C}_M$ is learnable by virtue of being polynomially explainable, then it is learnable even if examples of the target class are cluttered by a k-domination attribute noise, for any constant k.*

Proof. The proof is an extension of the arguments in Littlestone (1991) regarding robustness to attribute noise to the case of the infinite attribute model. It basically shows that the same algorithm works in the noisy case, only that the number of flipped relations affects the number of examples that the algorithm is required to see before it stops making mistakes.

3 From Theory to Practice

Several issues need to be addressed in order to exhibit the practicality of our learning framework. The first is the availability of a variety of RGFs that can be used to extract primitive visual patterns from data under different conditions and that are expressive enough so that a simple function defined on top of them is enough to discriminate an object. A basic assumption underlying this work is that this is not hard to do. In this work, we illustrate the approach by using simple edge detectors (clearly, too simple for a realistic situation). The second issue is the composition of complex, albeit simply defined, relations from primitive ones. This is crucial since it allows the representation of complex functions in terms of the instantiated relations by learning simple functional descriptions over their compositions. A language that supports composition of restricted families of conjunctions and can encode structural relations in images (e.g., above, to the left of ...) is discussed in Cumby and Roth (2000). The current work uses only general conjunctions and restricts only their size. Again, our working assumption is that even this simple representation is enough to generate a family of discriminating features.

Specifically, the above discussion amounts to assuming that it is (1) easy to extract simple relations over images—short vertical and horizontal edges in our case; (2) easy to extract simple monomials over these—each of our features represents the presence of some short conjunction of edges in the

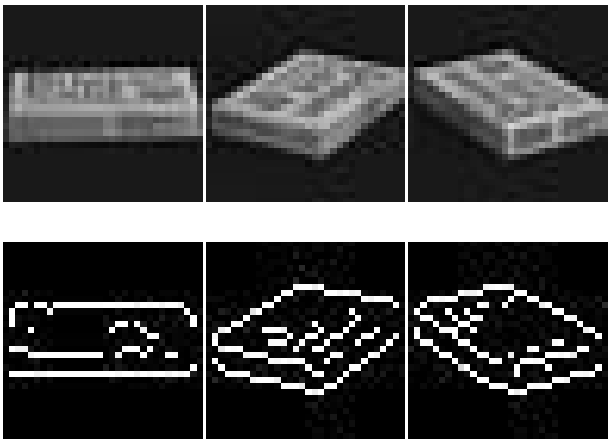

Figure 1: The short vertical and horizontal edges are extracted from an object image. These edges (and the conjunctions of them) represent the features of the object.

image; and (3) a linear threshold function (generalizing the simple disjunction in the theorems) over these features can be used to discriminate an object from other objects. We represent each object, therefore, as a linear function over the conjunctive features. In Figure 1, we exemplify the feature-based representation extracted for three objects when the features used are only short vertical and horizontal conjunctions (no conjunctions of those are used here). Our theory does not commit to this simplified and clearly insufficient representation. Nevertheless, such simple features capture sufficient information to recognize the 100 objects in the Columbia Object Image Library (COIL 100) database, as will be demonstrated.

Finally, the issue of the learnability of these representation is crucial in our approach. In learning situations in vision, the number of relations compositions (features) that could potentially affect each decision is very large, but typically, only a small number of them is actually relevant to a decision. Beyond correctness, a realistic learning approach therefore needs to be feature efficient (Littlestone, 1988) in that its learning complexity (the number of examples required for the learning algorithm to converge to a function that is a good discriminator) depends on the number of relevant features and not the global number of features in the domain. Equivalently, this can be phrased as the dependence of the true error on the error observed on the training data and the number of examples observed during training. For a feature-efficient algorithm, the number of training examples required in order to generalize well—to have true error that is not far from the error on the training data—is relatively small (Kivinen & Warmuth, 1995b).

A realistic learning approach in this domain should also allow the use of variable input size, for two reasons. First, learning is done in terms of relations that are generated from the image in a data-driven way, making it impossible, or impractical, for a learning approach to write explicitly, in advance, all possible relations and features. Similarly, dealing with a variable input size also allows an on-line learning scenario. Second, for computational efficiency purposes, since only a few of the many potential features are active in any instance, using a variable input size allows the complexity of evaluating the learning hypothesis on an instance to depend on the number of active features in the input rather than the total number in the domain.

Given that, the learning approach used in this work is the one developed within the SNoW learning architecture (Roth, 1998; Carlson et al., 1999). SNoW is specifically tailored for learning in domains in which the potential number of features taking part in decisions is very large but may be unknown a priori, as in the infinite attribute learning model (Blum, 1992). Specifically, as input, the algorithm receives labeled instances $< (x, l) >$, where an instance $x \in \{0, 1\}^{\infty}$ is presented as a list of all the active features in it and the label is a member of a discrete set of values (e.g., object identifiers). Given a domain instance (an image), a set of preexisting RGFs is evaluated on it and generates a collection of relations that are active in this image; these in turn may be composed to generate complex features, the elements of $\mathcal{M}$. A list (of unique identifiers) of active elements in $\mathcal{M}$ is presented to the learning procedure, and learning is done at this level. Specifically, given an image, the learning algorithm receives as input a description of it in terms of elements in $\mathcal{M}$ that are active in it. As required by the theorems already presented, at least some elements in this description should be relevant to the functional description of the target object, but many others may not be. The learning algorithm will quickly determine, by determining the weight of different features, the appropriate linear combination of features that can be used as a good discriminator.

We note that conceptually similar approaches have been developed by several researchers (Fleuret & Geman, 1999; Amit & Geman, 1999; Tieu & Viola, 2000; Mel & Fiser, 2000). In all cases, the approach is based on generating a fairly large number of features—typically generated as conjunctions of primitive feature—and hoping that a learning approach could learn a reliable discriminator as a function of these. These approaches made use of simple statistical methods (Mel & Fiser, 2000) or learning algorithms such as decision tree and AdaBoost (Fleuret & Geman, 1999; Tieu & Viola, 2000). Our approach differs in that (1) given some reasonable assumptions, we suggest a theoretical framework in which justifications can be developed and in which performance can be quantified as a function of the expressivity of the features used, and (2) it makes use of a different computational paradigm that we believe to be more appropriate in this domain. We use a feature-efficient learning method that provides the opportunity to learn high-order

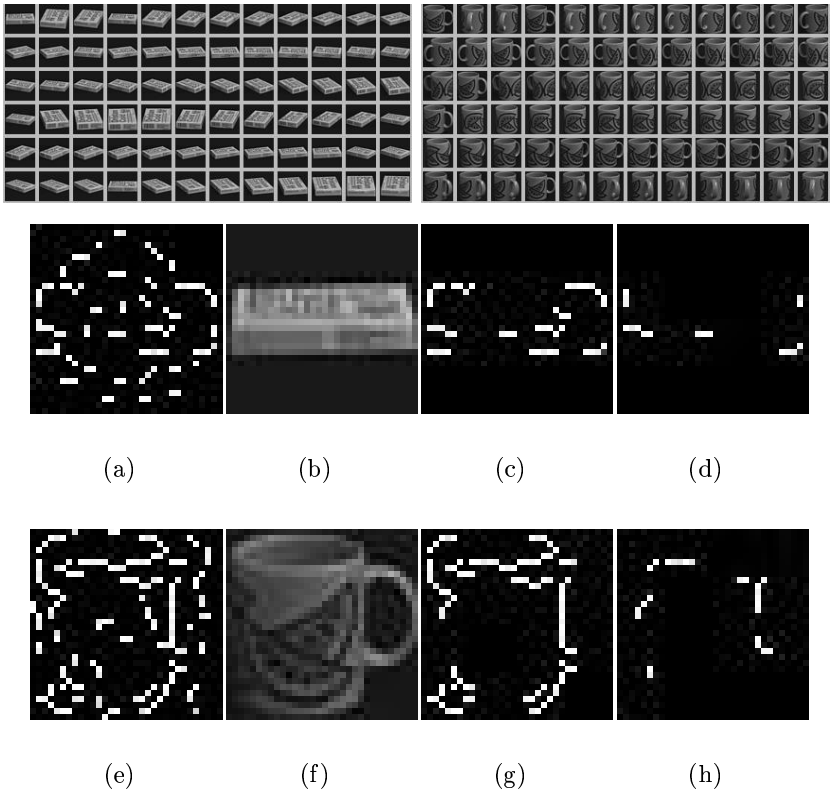

Figure 2: (Top) Images of the same object taken at different view points. We use SNoW to learn the representations of each object: one subnetwork of features for the pill box (b) and the other for mug (f). (a, e) respectively depict the features with top weights. Given a test image (b) and the learned representations (a, e), (c) shows the active features in the pill box subnetwork (a), and (d) shows the active features in the mug subnetwork (e). Given the test image (f) and the learned representations (a, e), (g) shows the active features in the mug subnetwork (e), and (h) shows the active features in the pill box subnetwork (a).

discriminators efficiently by increasing the dimensionality of the feature space (in a data-driven way) and still learn a linear function in the augmented feature space. This eliminates the need to perform explicit feature selection as in the other methods—it is done automatically and efficiently in the learning stage—or to deal with each feature separately (as in AdaBoost).

In Figure 2, we use two objects to illustrate the learned representation after training on a collection of examples, as well as the features in the learned representation that are active in a new example.

The images of a pill box object class and a mug object class taken at different view points (5 degrees apart) are shown at the top of figure 2. We extract small edges (of length 3) from each example and use these as the representation of each instance in training a SNoW system: one subnetwork for the pill box object class and the other for the mug object class (See also Figure 1 for examples of the input representation to SNoW.). Figures 2a and 2e depict the dominant features in the representations learned for the pill box object class and the mug object class (the weights of the feature are not shown; we show only those that have relatively high weights). Given the pill box test image shown in Figure 2b and the two learned subnetworks, Figure 2c shows the features of the test image that are active in the pill box subnetwork. Again, weights are not shown, but even in this way, it can be viewed as evidence that the test image belongs to the pill box object class. Figure 2d shows the features of the same test image that are active in the mug subnetwork. Here we see low evidence too that the test image belongs to the mug object class.

The mug test image shown in Figure 2f and the two learned subnetworks, Figures 2g and 2h, show features of the test image that are active in the mug and the pill box subnetworks. It seems clear, even with the simpler features shown here (only short edges) and without the weight information on the features, that the learned representations can discriminate images of objects taken from one target class from those taken from a different class of objects.

4 The SNoW Learning Architecture

The SNoW (Sparse Network of Winnows[5]) learning architecture is a sparse network of linear units over a common predefined or incrementally learned feature space. Nodes in the input layer of the network typically represent relations over the input instance and are being used as the input features. Each linear unit, called a target node, represents a concept of interest over the input. In the current application, target nodes represent a definition of an object in terms of the elements of $\mathcal{M}$—features extracted from the 2D image input. An input instance is mapped into a set of features active in it; this representation is presented to the input layer of SNoW and propagates to the target nodes. Target nodes are linked via weighted edges to (some of) the input features. Let $\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of features that are active in an example and are linked to the target node t. Then the linear unit corresponding to t is active iff

$$\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t,$$

[5] To winnow: to separate chaff from grain.

where w_i^t is the weight on the edge connecting the ith feature to the target node t, and θ_t is the threshold for the target node t.

Each SNoW unit may include a collection of subnetworks, one for each of the target relations but all using the same feature space. A given example is treated autonomously by each target unit; an example labeled t may be treated as a positive example by the t unit and as a negative example by the rest of the target nodes in its subnetwork. At decision time, a prediction for each subnetwork is derived using a winner-take-all policy. In this way, SNoW may be viewed as a multiclass predictor. In the application described here, we may have one unit with target subnetworks for all the target objects or we may define different units, each with two competing target objects.

SNoW's learning policy is on-line and mistake driven. Several update rules can be used within SNoW; the most successful and the only one used in this work is a variant of Littlestone's Winnow update rule (Littlestone, 1988), a multiplicative update rule that is tailored to the situation in which the set of input features is not known a priori, as in the infinite attribute model (Blum, 1992). This mechanism is implemented via the sparse architecture of SNoW. That is, (1) input features are allocated in a data-driven way—an input node for the feature i is allocated only if the feature i was active in any input vector—and (2) a link (i.e., a nonzero weight) exists between a target node t and a feature i if and only if i was active in an example labeled t.

One of the important properties of the sparse architecture is that the complexity of processing an example depends on only the number of features active in it, n_a, and is independent of the total number of features, n_t, observed over the lifetime of the system. This is important in domains in which the total number of features is very large but only a small number of them is active in each example.

The Winnow update rule has, in addition to the threshold θ_t at the target t, two update parameters: a promotion parameter $\alpha > 1$ and a demotion parameter $0 < \beta < 1$. These are being used to update the current representation of the target t (the set of weights w_i^t) only when a mistake in prediction is made. Let $\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of active features linked to the target node t. If the algorithm predicts 0 (that is, $\sum_{i \in \mathcal{A}_t} w_i^t \leq \theta_t$) and the received label is 1, the active weights in the current example are promoted in a multiplicative fashion:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \alpha \cdot w_i^t.$$

If the algorithm predicts 1 ($\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t$) and the received label is 0, the active weights in the current example are demoted:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \beta \cdot w_i^t.$$

All other weights are unchanged.

The key feature of the Winnow update rule is that the number of examples required to learn a linear function grows linearly with the number

n_r of relevant features and only logarithmically with the total number of features. Specifically, in the sparse model, the number of examples required before converging to a linear separator that separates the data (provided it exists) scales with $O(n_r \log n_a)$, where n_a is the number of active features observed. This property seems crucial in domains in which the number of potential features is vast but a relatively small number of them is relevant. Winnow is known to learn efficiently any linear threshold function (in general, the number of examples scales inversely with the margin (Littlestone, 1988) and to be robust in the presence of various kinds of noise and in cases where no linear-threshold function can make perfect classifications, while still maintaining its dependence on the number of total and relevant attributes (Littlestone, 1991; Kivinen & Warmuth, 1995a).

Once target subnetworks have been learned and the network is being evaluated, a decision support mechanism is employed, which selects the dominant active target node in the SNoW unit via a winner-take-all mechanism to produce a final prediction.

Figures 3, 4, and 5 provide more details on the SNoW learning architecture. Essentially, the SNoW learning architecture inherits its generalization properties from the update rule being used—the Winnow rule in this case—but there are a few differences worth mentioning relative to simply using the basic update rule. First, SNoW allows the use of a variable input size via the infinite attribute domain. Second, SNoW is more expressive than the basic Winnow rule. The basic Winnow update rule makes use of positive weights only. Standard augmentation, for example, via the duplication trick (Littlestone, 1988), is infeasible in high-dimensional spaces since they diminish the gain from using variable-size examples (since half of the features become active). More sophisticated approaches such as using the balanced version of Winnow apply only to the case of two classes, while SNoW is a multiclass classifier. Other extensions offered in SNoW relative to the standard update rule include an involved feature pruning method and a prediction confidence mechanism (Carlson, Rosen, & Roth, 2001).

$F = \mathcal{Z}^+ = \{0, 1, \ldots\}$	/* Set of potential features */
$T = \{t_1, \ldots t_k\} \subset F$	/* Set of targets */
$F_t \subseteq F$	/* Set of features linked to target t */
$t_{NET} = \{[(i, w_i^t)\colon\ i \in F_t], \theta_t\}$	/* The representation of the target t */
activation: $T \rightarrow \Re$	/* activation level of a target t */
$SNoW = \{t_{NET}\colon\ t \in T\}$	/* The SNoW Network */
$e = \{i_1, \ldots, i_m\} \subset F^m$	/* An example: a list of active features */

Figure 3: SNoW: Objects and notation.

Training Phase: SNoW-Train (SNoW, e)

Initially: $F_t = \phi$, for all $t \in T$.

For each $t \in T$

1. UpdateArchitecture (t, e)
2. Evaluate (t, e)
3. UpdateWeights (t, e)

Evaluation Phase: SNoW-Evaluation(SNoW, e)

For each $t \in T$

Evaluate (t, e)

MakeDecision (SNoW, e)

Figure 4: SNoW: Training and evaluation. Training is the learning phase in which the network is constructed and weights are adjusted. Evaluation is the phase in which the network is evaluated, given an observation. This is a conceptual distinction; in principle, one can run in on-line mode, in which training is done continuously, even when the network is used for evaluating examples.

Procedure Evaluate(t, e)

activation $= \sum_{i \in e} w_i^t$

Procedure UpdateWeights(t, e)

If $(activation(t) > \theta_t) \& (t \notin e)$ /* predicted positive on negative example */

for each $i \in e$: $w_t^i \leftarrow w_i^t \cdot \beta$

If $(activation(t) \leq \theta_t) \& (t \in e)$ /*predicted negative on a positive example*/

for each $i \in e$: $w_t^i \leftarrow w_i^t \cdot \alpha$

Procedure UpdateArchitecture(t, e)

If $t \in e$

for each $i \in e \backslash F_t$, set $w_i^t = w$ /* Link feature to target; set initial weight */

Otherwise: do nothing

Procedure MakeDecision(SNoW, e)

Predict winner = $\arg\max_{t \in T} activation(t)$ /* Winner-take-all Prediction */

Figure 5: SNoW: Main procedures.

The SNoW learning architecture has been used successfully on a variety of large-scale problems in the natural language domain (Roth, 1998; Munoz, Punyakanok, Roth, & Zimak, 1999; Golding & Roth, 1999) and only recently has been attempted on problems in the visual domain (Yang et al., 2000c).

Finally we note that, although not crucial to the main topic of this article, the SNoW learning architecture and the general approach of generating features for it could also be defended with neural plausibility in mind. First, SNoW represents objects in units that have only positive, excitatory connections corresponding to physiological facts that the firing rates of neurons cannot be negative; inhibitory effects of features arise from their occurrences in the representations of other objects. Moreover, a consequence of the weight update rule in the SNoW unit is that synapses do not change sign. Second, generating features as conjunctions of simple binary detectors (which can be SNoW units themselves) has been suggested before as an effective representation with potential biological plausibility (Fleuret & Geman, 1999; Ullman & Soloviev, 1999).

5 Discussion of Learning Methods

In this article, our learning approach is compared mostly to another linear learning approach: support vector machines (SVM). Below, we present the SVM method in some detail.

5.1 Support Vector Machines. SVM (Vapnik, 1995; Cortes & Vapnik, 1995) is a general-purpose learning method for pattern recognition and regression problems that is based on the theory of structural risk minimization. According to this principle, as described in section 2, a function that describes the training data well and belongs to a set of functions with low VC dimension will generalize well (that is, will guarantee a small, expected recognition error for the unseen data points) regardless of the dimensionality of the input space (Vapnik, 1995). Based on this principle, the SVM is a systematic approach to find a linear function (a hyperplane) that belongs to a set of functions of this form with the lowest VC dimension. Linear classifiers are used for computational purposes and, in addition, have the nice property that it is possible to quantify the VC dimension of this function class explicitly in terms of the minimal distance (margin) between positive and negative points (assuming the data is linearly separable).

For expressivity, SVMs provide nonlinear function approximations by mapping the input vectors into a high-dimensional feature space where a linear hyperplane that separates the data exists. It can also be extended to cases where the best hyperplane in the resulting high-dimensional space does not quite separate all the data points.

Given a set of samples $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_l, y_l)$ where $\mathbf{x}_i \in R^N$ is the input vector and $y_i \in \{-1, 1\}$ is its label, an SVM aims to find a separating hyperplane with the property that the distance it has from points of either

class (margin distance) is maximized. Vapnik (1995) shows that maximizing the margin distance is equivalent to minimizing the VC dimension and therefore contributes to better generalization. The problem of finding the optimal hyperplane is thus posed as a constrained optimization problem and solved using quadratic programming techniques. The optimal hyperplane, which determines the class label of a data point $\mathbf{x} \in R^N$, is of the form

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} y_i \alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b\right)$$

where $k(\cdot, \cdot)$ is a kernel function, used to map the original instance space to a high-dimensional space, b is a bias term, and sgn is the function that outputs $+1$ on positive inputs and -1 otherwise. Constructing an optimal hyperplane is equivalent to determining the nonzero α_i's. Sample vectors $\mathbf{x}_i$ that correspond to a nonzero α_i are called the support vectors (SVs) of the optimal hyperplane. The hope, when using this method, is to find a small number of SVs, thereby producing a compact classifier.

The use of kernel functions allows avoiding the need to blow up the dimensionality explicitly in order to reach a state in which the sample is linearly separable. If the kernel is of the form

$$k(\mathbf{x}, \mathbf{x}_i) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)$$

for some nonlinear function Φ: $R^N \rightarrow R^M$, the computation can be done in the original low-dimensional space rather than in the M-dimensional space, although the hyperplane is constructed in R^M. For a linear SVM, the kernel function is simply the dot product of vectors in the input space. Several kernel functions, such as polynomial functions and radial basis functions, have this property (Mercer's theorem) and can be used in nonlinear SVM, allowing the construction of a variety of learning machines, some of which coincide with classical architectures. However, this also results in a drawback since one needs to find the "right" kernel function when using SVMs. It is interesting to observe that although the use of kernel functions seems to be one of the advantages of SVMs from a theoretical point of view, many experimental studies have used linear SVMs, which were found to perform better than higher-level kernels (Pontil & Verri, 1998). This might be due to the fact that higher-level kernels imply in general worse generalization bounds, and thus require more examples to generalize well.

5.2 SNoW and SVMs. It is worthwhile to discuss the similarities and differences between the computational approaches we experiment with and to develop expectations to differences in the results.

At a conceptual level, both learning methods are very similar. They both search for a linear function that best separates the training data. Both are based on the same inductive principle: performing well on the training data

with a classifier of low expressivity would result in good generalization on data sampled from the same distribution.

Both methods work by blowing up the original instance space to a high-dimensional space and attempt to find a linear classifier in the new space. This gives rise to one significant difference between the methods. SVMs are a close relative of additive update algorithms like the perceptron (Rosenblatt, 1958; Freund & Schapire, 1998). For these algorithms, the dimensionality increase is done via the kernel functions and thus need not be done explicitly. The multiplicative update rule used in SNoW does not allow the use of kernels, and the dimensionality increase has to be done explicitly. Computationally, this could be significant. However, SNoW allows for the use of a variable input space, and since the feature space is sparse, it turns out that SNoW is actually more efficient than current SVM implementations. This advantage is significant when the examples are sparse (the number of active features in each example is small), and it disappears when there are many active features in each example, where the kernel-based methods are advantageous computationally. In addition, RGFs, which are the equivalent notion to kernels, could allow for more general transformations than those allowed by kernels (although in this work, we use conjunctions that are polynomial kernels).

A second issue has to do with the way the two methods determine the coefficients of the linear classifier and the implication this has for their generalization abilities. In SVMs, the weights are determined based on a global optimization criterion that aims at maximizing the margin, using a quadratic programming scheme. The generalization bounds achieved this way are related to those achieved by perceptron (Graepel, Herbrich, & Williamson, 2001). SNoW makes use of an on-line algorithm that attempts to minimize the number of mistakes on the training data; the loss function used to determine the weight update rule can be traced to the maximum entropy principle (Kivinen & Warmuth, 1995a). The implications are that while, in the limit, SVMs might find the optimal linear separator, SNoW has significant advantages in sparse spaces—those in which a few of the features are actually relevant (Kivinen & Warmuth, 1995a). We could expect, therefore, that in domains with these characteristics, if the number of training examples is limited, SNoW will generalize better (and in general will have better learning curves). In the limit, when sufficient examples are given, the methods will be comparable.

Finally, there is one practical issue: SVMs are binary classifiers, while SNoW can be used as a multiclass classifier. However, to get a fair comparison, we use SNoW here as a binary classifier as well, as described below.

6 View-Based Object Recognition

The appearance of an object is the combined effects of its shape, reflectance properties, pose, and illumination in the scene. While shape and reflectance

are intrinsic properties that do not change for a rigid object, pose and illumination vary from one scene to another. View-based recognition methods attempt to use data observed under different poses and illumination conditions to learn a compact model of the object's appearance; this, in turn, is used to resolve the recognition problem from view points that were not observed previously.

A number of view-based schemes have been developed to recognize 3D objects. Poggio and Edelman (1990) show that 3D objects can be recognized from the raw intensity values in 2D images (we call this representation here a pixel-based representation) using a network of generalized radial basis functions. Each radial basis generalizes and stores one of the example views and computes a weighting factor to minimize a measure of the error between the network's prediction and the desired output for each of the training examples. They argue and demonstrate that the full 3D structure of an object can be estimated if enough 2D views of the object are provided. This work has been extended to object categorization (Risenhuber & Poggio, 2000). (See also Edelman, 1999, for more details.)

Turk and Pentland (1991) demonstrate that human faces can be represented and recognized by "eigenfaces." Representing a face image as a vector of pixel values, the eigenfaces are the eigenvectors associated with the largest eigenvalues, which are computed from a covariance matrix of the sample vectors. An attractive feature of this method is that the eigenfaces can be learned from the sample images in pixel-based representation without any feature selection. The eigenspace approach has since been used in vision tasks from face recognition to object tracking. (Murase and Nayar, 1995; Nayar, Nene, & Murase, 1996) develop a parametric eigenspace method to recognize 3D objects directly from their appearance. For each object of interest, a set of images in which the object appears in different poses is obtained as training examples. Next, the eigenvectors are computed from the covariance matrix of the training set. The set of images is projected to a low-dimensional subspace spanned by a subset of eigenvectors in which the object is represented as a manifold. A compact parametric model is constructed by interpolating the points in the subspace. In recognition, the image of a test object is projected to the subspace, and the object is recognized based on the manifold it lies on. Using a subset of COIL 100, they show that 3D objects can be recognized accurately from their appearances in real time.

In contrast to these algebraic methods, general-purpose learning methods such as SVMs have also been used for this problem. Schölkopf (1997) applies SVMs to recognize 3D objects from 2D images and demonstrate the potential of this approach in visual learning. Pontil and Verri (1998) also use SVMs for 3D object recognition and experimented with a subset of the COIL 100 data set. Their training set consisted of 36 images (one for every 10 degrees) for each of the 32 objects they chose, and the test sets consist of the remaining 36 images for each object. For 20 random selections of 32

objects from the COIL 100, their system achieves perfect recognition rate (but see the comments on that in section 7). Recently, Roobaert and M. Van Hulle (1999) also used a subset of the COIL 100 database to compare the performance of SVMs with different pixel-based input representations.

Instead of using the whole appearance of an object for object recognition, several methods have used local features in visual learning. Le Cun et al. (1995) apply a convolutional neural network with local features to handwritten digit recognition, with very good results. They also demonstrate that their learning method is able to extract salient local features from example images without complicated and elaborated algorithms. The idea of estimating joint statistics of local features has been used in recent work. Amit and Geman (1999) use conjunction of edges as local features of image and apply tree classifiers to recognize handwritten digits. Schneiderman and Kanade (1998) use naive Bayes classifier to model the joint distribution of features from face images and use the learned model for face detection with success. Viola and colleagues (De Bonet & Viola, 1998; Rikert, Jones, & Viola, 1999; Tieu & Viola, 2000) assume that the appearance of an object in one image is generated by a sparse set of visual local causes (i.e., features). Their method computes a large set of selective features from examples to capture local causal structure and applies a variation of AdaBoost (Freund & Schapire, 1997) with gaussian models to learn a hypothesis of an object. Other examples include object recognition using high-dimensional iconic representations (Rao & Ballard, 1995), multidimensional histograms (Schiele, 2000), local curve features (Nelson & Selinger, 1998), conjunction of local features (Mel, 1997), SVMs with local features using wavelets (Papageorgiou & Poggio, 2000), local principal component analysis (Penev & Atick, 1996) and independent component analysis (Donato, Bartlett, Hager, Ekman, & Sejnowski, 1999).

The current work is most related conceptually to a collection of other works that make use of local features and their joint statistics (De Bonet & Viola, 1998; Rikert et al., 1999; Nelson & Selinger, 1998; Amit & Geman, 1999; Schiele, 2000). As in these works, the key assumption underlying our work is that objects can be recognized (or discriminated) using simple representations in terms of syntactically simple relations over the raw image. Based on these assumptions, this work provides a learning theory account for the problem of object recognition within the PAC model of learnability. Moreover, the computational approach developed and supported here is different from previous approaches and more suitable, we believe, to realistic visual learning situations.

Although the number of these simple relations could be huge, at the basis of our computational approach is the belief that only a few of them are actually present in each observed image and a fairly small number of those observed is relevant to discriminating an object. Under these assumptions, our framework has several theoretical advantages that we described in the previous sections. For our framework to contribute to a practical solution,

there also needs to be a computational approach that is able to learn efficiently (in terms of both computation and number of examples) in the presence of a large number of potential explanations. Our evaluation of the theoretical framework makes use of the SNoW learning architecture (Roth, 1998; Carlson et al., 1999), which is tailored for these kind of tasks.

Next we use the COIL 100 data set for and quantitative experimental evaluation.

7 Experimental Evaluation

We use the COIL 100 database in all the experiments below (COIL is available on-line at www.cs.columbia.edu/CAVE). The data set consists of color images of 100 objects where the images of the objects were taken at pose intervals of 5 degrees, for 72 poses per object. The images were also normalized such that the larger of the two object dimensions (height and width) fits the image size of 128×128 pixels. Figure 6 shows the images of the 100 objects taken in frontal view (i.e., zero pose angle). The 32 highlighted objects in Figure 6 are considered more difficult to recognize in Pontil and Verri (1998); we use all 100 objects, including these in our experiments. Each color image is converted to a gray-scale image of 32×32 pixels for our experiments.

7.1 Ground Truth of the COIL 100 Data Set. At first glance, it seems difficult to recognize the objects in the COIL data set because it consists of a large number of objects with varying pose, texture, shape, and size. Since each object has 72 images of different poses (5 degrees apart), many view-based recognition methods use 36 (10 degrees apart) of them for training and the remaining images for testing. However, it turns out that under these dense sampling conditions, the recognition problem is not difficult (even when only gray-level images are used). In this case, instances that belong to the same object are very close to each other in the image space (where each data point represents an image of an object in a certain pose). We verified this by experimenting with a simple nearest-neighbor classifier (using the Euclidean distance), resulting in an average recognition rate of 98.50% (54 errors out of 3,600 tests). Figure 7 shows some of the objects misclassified by the nearest-neighbor method.

In principle, one may want to avoid using the nearest-neighbor method since it requires a lot of memory for storing templates and its recognition time complexity is high. The goal here is simply to show that this method is comparable to the complex SVM approaches (Pontil & Verri, 1998; Roobaert & Hulle, 1999) for the case of dense sampling. Therefore, the recognition problem is not appropriate for comparison among different methods.

It is interesting to see that the pairs of the objects on which the nearest-neighbor method misclassified have similar geometric configurations and similar poses. A close inspection shows that most of the recognition errors are made between the three packs of chewing gums, bottles, and cars. Other

Figure 6: Columbia Object Image Library (COIL 100) consists of 100 objects of varying poses 5 degrees apart. The objects are shown in row order; the highlighted ones are those considered more difficult to recognize.

dense sampling cases are easier for this method. Consequently, the set of selected objects in an experiment has direct effects on the recognition rate. This needs to be taken into account when evaluating results that use only a subset of the 100 objects (typically 20 to 30) from the COIL data set for experiments. Table 1 shows the recognition rates of nearest-neighbor classifiers in several experiments in which 36 poses of each object are used for templates and the remaining 36 poses are used for tests.

Given this baseline experiment, we decided to perform our experimental comparisons in cases in which the number of views of objects available in training is limited. Some of our preliminary results were presented in Yang, Roth, and Ahuja (2000a, 2000b).

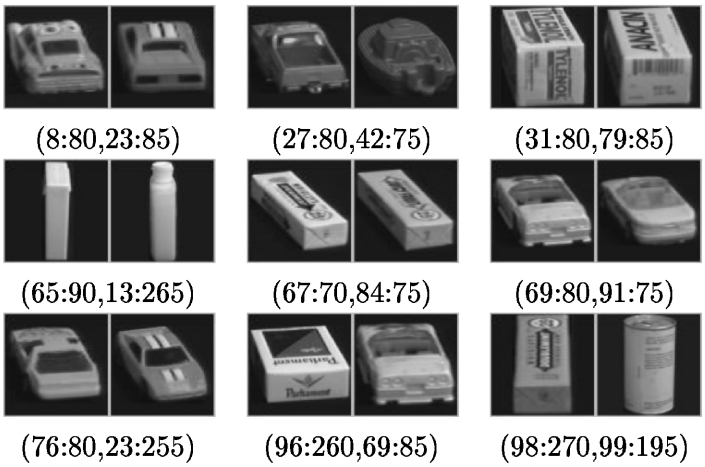

Figure 7: Mismatched objects using the nearest-neighbor method. $(x : a, y : b)$ means that object x with view angle a is recognized as object y with view angle b. It shows some of the 54 errors (out of 3600 test samples) made by the nearest-neighbor classifier when there are 36 views per object in the training set.

Table 1: Recognition Rates of Nearest-Neighbor Classifier.

	30 Objects Randomly Selected from COIL	32 Objects Shown in Figure 6 Selected by Pontil and Verri (1998)	The 100 Objects in COIL
Errors/tests	14/1080	46/1152	54/3600
Recognition rate	98.70%	96.00%	98.50%

7.2 Experiment Setups. Applying SNoW to 3D object recognition requires specifying the architecture used and the representation chosen for the input images. To perform object recognition, we associate a target unit with each target object. This target learns a definition of the object in terms of the input features extracted from the image. We could define either a single SNoW unit that contains target subnetworks for all 100 different target objects or different units, each with several (e.g., two) competing target objects. Statistically, this approach is advantageous (Hastie & Tibshirani, 1998), although it clearly requires a lot more computation. The architecture selected affects the training time, where learning a definition for object a makes use of negative examples of other objects that are part of the same unit. More important, it makes a difference in testing; rather than two competing objects for a decision, there may be a hundred. The chances for a spurious mistake caused by an incidental view point are clearly much higher. It also

Table 2: Experimental Results of Three Classifiers Using the 100 Objects in the COIL-100 Data Set.

	Number of Views per Object			
	36	18	8	4
	3600 Tests	5400 Tests	6400 Tests	6800 Tests
SNoW	95.81%	92.31%	85.13%	81.46%
Linear SVM	96.03	91.30	84.80	78.50
Nearest neighbor	98.50	87.54	79.52	74.63

has significant advantages in terms of space complexity and the appeal of the evaluation mode.

SVMs are two-class classifiers that for a c-class pattern recognition problem usually need to train $\frac{c(c-1)}{2}$ binary classifiers. Since we compare the performance of the proposed SNoW-based method with SVMs, in order to maintain a fair comparison we have to perform it in the one-against-one scheme. That is, we use SNoW units of size two. To classify a test instance, tournament-like pair-wise competition between all the machines is performed, and the winner determines the label of the test instance. Table 2 shows the recognition rates of the SVM- and SNoW-based methods using the one-against-one scheme. (That is, we trained $\binom{100}{2} = 4950$ classifiers for each method and evaluated $99(50 + 25 + 12 + 6 + 3 + 2 + 1)$ classifiers on each test instance.)

7.3 Results Using Pixel-Based Representation. Table 2 shows the recognition rates of the SNoW-based method, the SVM-based method (using linear dot product for the kernel function), and the nearest-neighbor classifier using the COIL 100 data set. The important parameter we vary here is the number of views (v) observed during training; the rest of the views ($72 - v$) are used for testing.

The experimental results show that the SNoW-based method performs as well as the SVM-based method when many views of the objects are present during training and outperforms the SVM-based method when the number of views is limited. Although it is not surprising to see that the recognition rate decreases as the number of views available during training decreases, it is worth noticing that both SNoW and SVM are capable of recognizing 3D objects in the COIL 100 data set with satisfactory performance if enough views (e.g., more than 18) are provided. Also they seem to be fairly robust even if only a limited number of views (e.g., 8 and 4) are used for training; the performance of both methods degrades gracefully.

An additional potential advantage of the SNoW architecture is that it does not learn discriminators but rather can learn a representation for each object,

Table 3: Recognition Rates of SNoW Using Two Learning Paradigms.

	Number of Views per Object			
SNoW	36	18	8	4
One-against-one	95.81%	92.31%	85.13%	81.46%
One-against-all	90.52	84.50	81.85	76.00

which can then be used for prediction in the one-against-all scheme or to build hierarchical representations. See Figure 2 for examples. However, as is shown in Table 3, this implies a significant degradation in the performance. Finding a way to make better predictions in the one-against-all scheme is one of the important issues for future investigation, to exploit the advantages of this approach better.

7.4 Results Using Edge-Based Representation. For each 32×32 edge map, we extract horizontal and vertical edges (of length at least 3 pixels) and then encode as our features conjunctions of two of these edges. The number of potential features of this sort is $\binom{2048}{2} = 2{,}096{,}128$. However, only an average of 1822 of these is active for objects in the COIL 100 data set. To reduce the computational cost, the feature vectors were further pruned, and only the 512 most frequently occurring features were retained in each image.

Table 4 shows the performance of the SNoW-based method when conjunctions of edges are used to represent objects. As before, we vary the number of views of an object (v) during training and use the rest of the views ($72 - v$) for testing. The results indicate that conjunctions of edges provide useful information for object recognition and that SNoW is able to learn very good object representations using these features. The experimental results also exhibit the relative advantage of this representation when the number of views per object is limited.

7.5 Simple Occlusion. This section presents some preliminary studies of object recognition in the presence of occlusion. Our current modeling of occlusion is fairly simplistic relative to studies such as Nelson and Selinger (1998) and Schiele (2000). However, given that our theoretical paradigm supports recognition in the presence of occlusion, we wanted to experiment with this in the current setting.

We select a set of 10 objects[6] from the COIL 100 data set and add in artificial occlusions for experiments. In the data set, each object has 36 images (10 degrees apart) for training and the remaining 36 images for tests (also 10

[6] More specifically, the objects are selected from the set of objects on which the nearest-neighbor classifier makes the most mistakes: objects 8, 13, 23, 27, 31, 42, 65, 78, 80, 91.

Table 4: Experimental Results of Three Classifiers Using the 100 Objects in the COIL-100 Data Set.

	Number of Views per Object			
	36	18	8	4
	3600 Tests	5400 Tests	6400 Tests	6800 Tests
SNoW with conjunction of edges	**96.25%**	**94.13%**	**89.23%**	**88.28%**
SNoW with intensity values	95.81	92.31	85.13	81.46
Linear support vector machine	96.03	91.30	84.80	78.50
Nearest neighbor	98.50	87.54	79.52	74.63

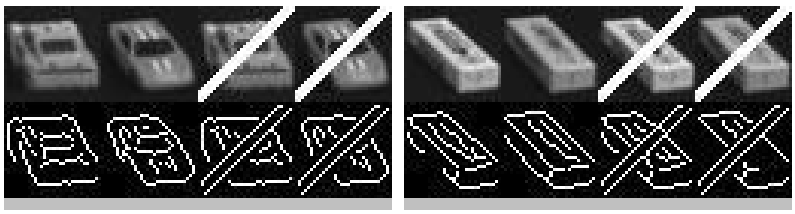

Figure 8: Object images with and without occlusion as well as their edge maps.

degrees apart). The object images are occluded by a strip controlled by four parameters (α, p, l, g), where α denote the angle of the strip, p denote the percentage of occluded image area, l denote the location of the center of the strip, and g denote the intensity values of the strip. Figure 8 shows some object images and the occluded object images for $\{\alpha, p, l, g\} = \{45^\circ, 15\%, (16, 16), 0\}$. Our trained SNoW classifier is tested against this data set using the edge-based representation. Table 5 shows the experimental results with and without occlusions on this set of 10 objects with 36 views. The recognition performance degrades only slightly from 92.03% to 88.78%. Note that the objects are those on which the nearest-neighbor classifier makes the most mistakes.

Although our theoretical framework supports noise tolerance, it is clear that the limited expressivity of the features used in this experimental study

Table 5: Experimental Results of SNoW Classifier on Occluded Images with 36 Views per Object.

	Recognition Rate Without Occlusion	Recognition Rate With Occlusion
SNoW	92.03%	88.78%

limits the ability to tolerate occlusion; we nevertheless find our results promising.

8 Conclusion

The main contribution of this work is proposing a learning framework for visual learning and exhibiting its feasibility. In this approach, learnability can be rigorously studied without making assumptions on the distribution of the observed objects; instead, via the PAC model, the performance of the learned hypothesis naturally depends on its prior experience. An important feature of the approach is that learning is studied not directly in terms of the raw data but rather with respect to intermediate representations extracted from it and can thus be quantified in terms of the ability to generate expressive intermediate representations. In particular, it makes explicit the requirements from these representations to allow learnability. We believe that research in vision should concentrate on the study of these intermediate representations.

We evaluated the approach and demonstrated its feasibility in a large-scale experiment in the context of learning for object recognition. Our experiments allowed us also to perform a fair comparison between two successful and related learning methods and study them in the context of object recognition. We have illustrated our approach in a large-scale experimental study in which we use the SNoW learning architecture to learn representations for the objects in COIL 100. Although it is clear that object recognition in isolation is not the ultimate goal, this study shows the potential of this computational approach as a basis for studying and supporting more realistic visual inferences.

We note that for a fair comparison among different methods, we have used pixel-based presentation in the experiments. The experimental results suggest that the edge-based representation used is more effective and robust and should be the starting point for future research. There is no question that the RGFs used in this work are not general enough to support more challenging recognition problems; the intention was merely to exhibit the general approach. We believe that pursuing the direction of using complex intermediate representations will benefit future work on recognition and, in particular, robust recognition under realistic conditions.

The framework developed here is very general. The explanations used as features by the learning algorithm can represent a variety of computational processes and information sources that operate on the image. They can depend on local properties of the image, the relative positions of primitives in the image, and even external information sources or context variables. Thus, the theoretical support given here applies also to an intermediate learning stage in a hierarchical process. In order to generate the explanations efficiently, this work assumes that they are syntactically simple in terms of the raw image. However, the explanation might as well be syntactically

simple in terms of previously learned or inferred predicates, giving rise to a hierarchical representation.

We believe that the key future research question suggested by this line of work is that of incorporating more visual knowledge into instantiations of this framework and, in particular, using it to generate better explanations.

Acknowledgments

D. R. is supported by National Science Foundation grants IIS-9801638, IIS-9984168, and ITR-IIS-0085836, M-S. Y. was supported by a Ray Ozzie Fellowship and Office of Naval Research (ONR) grant N00014-96-1-0502 and currently is with Honda Fundamental Research Labs, Mountain View, California. N. A. is supported by ONR grant N00014-96-1-0502.

References

Amit, Y., & Geman, D. (1999). A computational model for visual selection. *Neural Computation, 11*(7), 1691–1715.

Angluin, D. (1992). Computational learning theory: Survey and selected bibliography. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing* (pp. 351–369). Victoria, B.C., Canada.

Blum, A. (1992). Learning Boolean functions in an infinite attribute space. *Machine Learning*, *9*(4), 373–386.

Carlson, A., Cumby, C., Rosen, J., & Roth, D. (1999). *The SNoW learning architecture* (Tech. Rep. No. UIUCDCS-R-99-2101). Urbana, IL: UIUC Computer Science Department.

Carlson, A. J., Rosen, J., & Roth, D. (2001). Scaling up context sensitive text correction. In *Proceedings of the Thirteenth Innovative Applications of Artificial Intelligence Conference*. Seattle, WA.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*(3), 273–297.

Cumby, C., & Roth, D. (2000). Relational representations that facilitate learning. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning* (pp. 425–434). Breckinridge, CO.

De Bonet, J., & Viola, P. (1998). Texture recognition using a non-parametric multi-scale statistical model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 641–647). Santa Barbara, CA.

Donato, G., Bartlett, M. S., Hager, J. C., Ekman, P., & Sejnowski, T. J. (1999). Classifying facial actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(10), 974–989.

Edelman, S. (1993). On learning to recognize 3-D objects from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(8), 833–837.

Edelman, S. (1999). *Representation and recognition in vision*. Cambridge, MA: MIT Press.

Fleuret, F., & Geman, D. (1999). Graded learning for object detection. In *Proceedings of the IEEE Workshop on Statistical and Computational Theories of Visions*. Fort Collins, CO.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Freund, Y., & Schapire, R. (1998). Large margin classification using the perceptron algorithm. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory* (pp. 209–217). Madison, WI.

Golding, A. R., & Roth, D. (1999). A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, *34*(1–3), 107–130. (Special Issue on Machine Learning and Natural Language.)

Graepel, T., Herbrich, R., & Williamson, R. C. (2001). From margin to sparsity. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13* (pp. 210–216). Cambridge, MA: MIT Press.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems, 10* (pp. 507–513). Cambridge, MA: MIT Press.

Haussler, D. (1992). Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computer*, *100*(1), 78–150.

Haussler, D., Kearns, M., Littlestone, N., & Warmuth, M. K. (1988). Equivalence of models for polynomial learnability. In *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 42–55). Cambridge, MA.

Kearns, M., & Li, M. (1993). Learning in the presence of malicious error. *SIAM Journal of Computing*, *22*(4), 807–837.

Kearns, M. J., Schapire, R. E., & Sellie, L. M. (1994). Toward efficient agnostic learning. *Machine Learning*, *17*(2/3), 115–142.

Kivinen, J., & Warmuth, M. K. (1995a). Additive versus exponentiated gradient updates for linear prediction. In *Proceedings of the Annual ACM Symposium on Theory of Computing* (pp. 209–218). Las Vegas, NV.

Kivinen, J., & Warmuth, M. K. (1995b). The perceptron algorithm vs. Winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proceedings of 8th Annual Conference on Computational Learning Theory* (pp. 289–296). Santa Cruz, CA.

Kushilevitz, E., & Roth, D. (1996). On learning visual concepts and DNF formulae. *Machine Learning*, *24*(1), 65–85.

Le Cun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., & Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. In *Proceedings of International Conference on Artificial Neural Networks* (pp. 53–60). Paris, France.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Littlestone, N. (1991). Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory* (pp. 147–156). Santa Cruz, CA.

Mel, B. W. (1997). Seemore: Combing color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation*, *9*, 777–804.

Mel, B. W., & Fiser, J. (2000). Minimizing binding errors using learned conjunctive features. *Neural Computation*, *12*, 247–278.

Munoz, M., Punyakanok, V., Roth, D., & Zimak, D. (1999). A learning approach to shallow parsing. In *EMNLP-VLC'99, the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (pp. 168–178). College Park, MD.

Murase, H., & Nayar, S. K. (1995). Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, *14*, 5–24.

Nayar, S. K., Nene, S. A., & Murase, H. (1996). Real-time 100 object recognition system. In *Proceedings of IEEE International Conference on Robotics and Automation*. Minneapolis, MN.

Nelson, R. C., & Selinger, A. (1998). Large-scale tests of a keyed, appearance-based 3-D object recognition system. *Vision Research*, *38*(15/16), 2469–2488.

Papageorgiou, C., & Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, *38*(1), 15–33.

Penev, P., & Atick, J. (1996). Local feature analysis: A general statistical theory for object representation. *Network: Computation in Neural Systems*, *7*(3), 477–500.

Poggio, T., & Edelman, S. (1990). A network that learns to recognize 3D objects. *Nature*, *343*, 263–266.

Pontil, M., & Verri, A. (1998). Support vector machines for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(6), 637–646.

Rao, R. P. N., & Ballard, D. H. (1995). An active vision architecture based on iconic representations. *Artificial Intelligence*, *78*(1–2), 461–505.

Rikert, T., Jones, M., & Viola, P. (1999). A cluster-based statistical model for object detection. In *Proceedings of the Seventh IEEE International Conference on Computer Vision* (pp. 1046–1053). Kerkyra, Greece.

Risenhuber, M., & Poggio, T. (2000). Models of object recognition. *Nature Neuroscience*, *3*, 1199–1204.

Roobaert, D., & Hulle, M. V. (1999). View-based 3D object recognition with support vector machines. In *IEEE International Workshop on Neural Networks for Signal Processing* (pp. 77–84). Madison, WI.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–407.

Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 806–813). Madison, WI.

Schiele, B. (2000). Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, *36*(1), 31–50.

Schneiderman, H., & Kanade, T. (1998). Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 45–51). Santa Barbara, CA.

Schölkopf, B. (1997). *Support vector learning*. Munich: Oldenbourg Verlag.

Shvaytser, H. (1990). Learnable and nonlearnable visual concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(5), 459–466.

Tieu, K., & Viola, P. (2000). Boosting image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (Vol. 1, pp. 228–235). Hilton Head, SC.

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3*(1), 71–86.

Ullman, S., & Soloviev, S. (1999). Computation of pattern invariance in brain-like structures. *Neural Networks*, *12*, 1021–1036.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.

Vapnik, V. N. (1982). *Estimation of dependences based on empirical data*. New York: Springer-Verlag.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

Yang, M.-H., Roth, D., & Ahuja, N. (2000a). Learning to recognize 3D objects with SNoW. In *Proceedings of the Sixth European Conference on Computer Vision* (Vol. 1, pp. 439–454). Dublin, Ireland.

Yang, M.-H., Roth, D., & Ahuja, N. (2000b). View-based 3D object recognition using SNoW. In *Proceedings of the Fourth Asian Conference on Computer Vision* (Vol. 2, pp. 830–835). Denver, CO.

Yang, M.-H., Roth, D., & Ahuja, N. (2000b). A SNoW-based face detector. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12* (pp. 855–861). Cambridge, MA: MIT Press.

Received August 23, 2000; accepted July 27, 2001.